szkolenia pod drzewem



Scrum Podstawy

1.26.15

© 2006–2011 Tomasz Włodarek. Materiał udostępniony na podstawie licencji Creative Commons (by-nc-nd)

APLN
Agile Project Leadership Network

Prowadzący

Tomasz Włodarek

PSMT, PSM, CSM, CSPO, CSP, itp. itd. ☺

Obecnie:

niezależny trener, coach, konsultant

Do niedawna:

Senior Project Manager, Sabre Holdings, Kraków

Wcześniej:

Product Team Manager, Motorola, Kraków

(Senior) Software Engineer, Motorola, Kraków

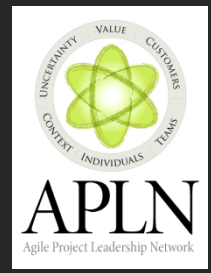
Game Developer, Game Co-Author, Optimus Nexus, Gdańsk

Pełny profil zawodowy:

www.linkedin.com/in/wlodarek

www.poddrzewem.pl

tomek@poddrzewem.pl





- Doświadczenie zawodowe Tomka obejmuje 14 lat pracy w projektach związanych z produkcją oprogramowania na stanowiskach inżynierskich (8) i kierowniczych (6). W codziennej pracy ma do czynienia z zagadnieniami związanymi z prowadzeniem zespołów produktowych i projektów w granicach kilkunastu osobolat, zgodnie z procesami empirycznymi (agile) ale także zdefiniowanymi, kaskadowymi procesami, w tym opartymi o model SEI CMMI®.
- Jest praktykiem, entuzjastą i orędownikiem lekkich metod w produkcji oprogramowania oraz aktywnym członkiem społeczności agile'owej.
- Specjalizuje się w zagadnieniach z zakresu produkcji oprogramowania, w szczególności definiowaniu i dostosowywaniu procesów wytwórczych oprogramowania oraz formowaniu, szkoleniach i coachingu zespołów deweloperskich.

Inicjatywa Pod Drzewem

- Aktywne propagowanie pragmatycznych, zwinnych metod produkcji oprogramowania (agile, lean)
- Szkolenia i warsztaty z zakresu zarządzania projektami w sposób empiryczny i adaptacyjny
- Formowanie, mentoring i coaching zespołów projektowych
- Doradztwo w zakresie wdrażania podejścia empirycznego
- Doradztwo w zakresie migracji lub definiowania adaptacyjnych procesów produkcji
- Doradztwo w zakresie definiowania struktur i procesów w organizacjach IT
- **Projekty szkoleniowo-doradcze realizowane m.in. dla:**

  ABG S.A., Anixe Polska, Apriso Polska, Asseco Business Solutions S.A., ATSI S.A., BLStream, CCA Europe, CD Projekt RED, Copi S.A., GE Money Bank S.A., Getin Bank S.A., Hurra Communications, Logintrans, Nokaut.pl, Quantum Software S.A., QXL Poland (Grupa Allegro), RST, Sterkom, Travelplanet.pl, TRW Polska, TVN, Volantis Systems, VSoft S.A.

Scrum Podstawy

Celem szkolenia jest nabycie przez uczestników podstawowej wiedzy na temat metody Scrum i korzyści wynikających z jej zastosowania w procesie wytwarzania oprogramowania.

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

Program szkolenia

- Wstęp do Agile Software Development
- Wstęp do Scrum'a, geneza, wartości
- Role i zakresy odpowiedzialności zespołu scrumowego
- Minimalistyczne narzędzia
- Przebieg iteracji, praktyki i ceremonia
- Monitorowanie przebiegu iteracji
- Analiza i adaptacja
- Prędkość zespołu
- Monitorowanie przebiegu projektu
- Sesja pytań i odpowiedzi

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



podejście tradycyjne

(z braku lepszego określenia)

plan–wykonanie

Planowanie detaliczne i życzeniowe

Wiarygodność planu jest zależna od skali, stopnia niepewności i złożoności przedsięwzięcia

Przygotowanie i utrzymanie planu jest kosztowne

Odstępstwo od planu jest kosztowne

Zdefiniowane i powtarzalne procesy

Silnie wyodrębnione fazy produkcji

Struktura organizacji sprzyja powstawaniu silosów kompetencyjnych

Długie cykle produkcyjne, późna kapitalizacja

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

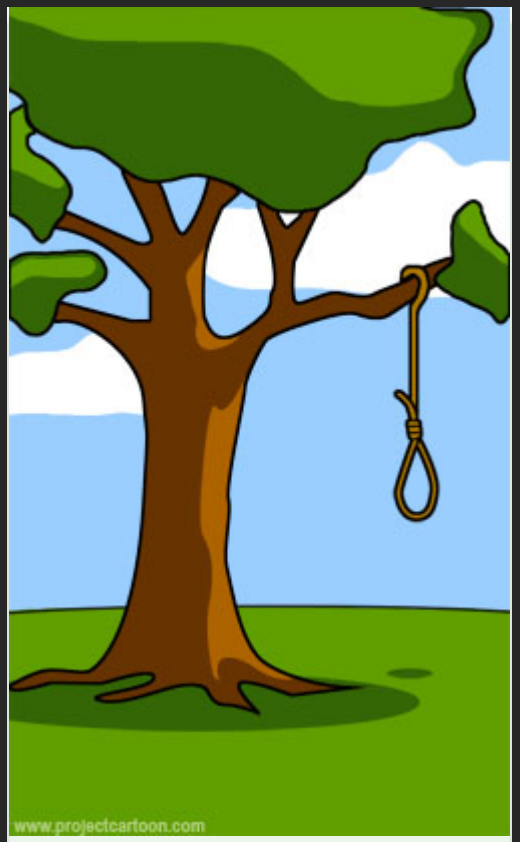


Co mi zrobisz, jak mnie złapiesz? reż. Stanisław Bareja (1978)

*



Oto czego naprawdę potrzebował klient...



...a oto co zostało mu dostarczone

więcej wariantów pod adresem: www.projectcartoon.com



podejście adaptacyjne

zwane również empirycznym (z braku lepszego określenia)

wizja–eksploracja

Nieprecyzyjne wymagania wymuszają ciągłą weryfikację kierunku rozwoju

Podstawowym kryterium oceny jest stopniowo rozbudowywany produkt

Zmiana jest oczekiwanym i akceptowanym elementem procesu

Planowanie jest adaptacyjne, a procesy empiryczne

Ciągła optymalizacja procesu i zwiększanie efektywności jest priorytetem

Przenikające się fazy produkcji

Struktura organizacji oparta o interdyscyplinarne, samoorganizujące się zespoły

Krótkie cykle produkcyjne, możliwa wczesna kapitalizacja

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

postulat agile software development*

software ≠ produkcja przemysłowa

software=tworzenie nowego produktu (projektowanie)

*agile (pol. sprawny, zwinny, rzutki, zwrotny, żwawy) – najczęściej stosowane polskie określenia w kontekście produkcji oprogramowania: lekkie, zwinne [techniki, metody, metodyki]

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

argumenty wspierające postulat

- Szybko zmieniające się technologie (SW/HW)
- Ogromna złożoność, kompleksowe zależności
- Innowacyjność wymuszona silną konkurencją
- Zmieniające się wymagania
- Produkcja oprogramowania to praca z ludźmi (kreatywnymi, twórczymi, o wysokich kwalifikacjach)
- Stosunkowa łatwość wprowadzania zmian
- Stosunkowo niskie koszty wprowadzania zmian
- **W większości przypadków powstaje nowy produkt o unikalnej funkcjonalności będącej odpowiedzią na specyficzne potrzeby klienta**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

Odkrywamy coraz doskonalsze sposoby rozwijania oprogramowania, tworząc je i pomagając innym to robić. Działając w ten sposób nauczyliśmy się cenić:

manifest agile*

ludzi i współpracę bardziej niż procesy i narzędzia
działające oprogramowanie bardziej niż wyczerpującą dokumentację
współpracę z klientem bardziej niż formalne pertraktacje
reagowanie na zmiany bardziej niż podążanie za planem

*http://www.agilemanifesto.org

wysokiej jakości kod bardziej niż pospolite dziadostwo*

* Robert (Uncle Bob) Martin

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



w naszej pracy kierujemy się regułami –

- Najistotniejsza jest dla nas satysfakcja klienta. Osiągamy ją przez wczesne (w cyklu życia projektu) i ciągłe dostarczanie wartościowego oprogramowania.
- Zmiany wymagań przyjmujemy są z aprobatą, nawet jeśli występują późno w cyklu życia projektu. Procesy zwinne zaprzęgają zmianę do utrzymania przewagi konkurencyjnej naszych klientów.
- Dostarczamy działające oprogramowanie często, w odstępach od kilku tygodni do kilku miesięcy. Preferujemy krótsze okresy czasu.
- Osoby reprezentujące biznes muszą pracować razem z zespołem produkcyjnym – dzień w dzień – przez cały okres realizacji projektu.
- Realizujemy projekty opierając się na motywacji osób biorących w nich udział. Udostępniamy potrzebne im zasoby i ufamy, że ich praca zostanie wykonana.
- Najbardziej skutecznym i wydajnym sposobem przekazywania informacji do i w ramach zespołu projektowego jest bezpośrednia rozmowa.

*http://www.agilemanifesto.org/principles.html

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



w naszej pracy kierujemy się regułami –

- Działające oprogramowanie jest podstawową miarą postępów prac.
- Procesy zwinne promują zrównoważone tempo rozwijania oprogramowania. Sponsorzy, zespół projektowy i przyszli użytkownicy powinni być w stanie utrzymywać stałe tempo bez końca.
- Bezustanne poświęcanie uwagi utrzymywaniu najwyższych standardów inżynieryjnych i odpowiedniej architektury systemu skutkuje większą sprawnością (zwinnością).
- Prostota – sztuka maksymalizowania pracy, której nie trzeba wykonywać – jest niezbędna.
- Najlepsze możliwe: architektura, wymagania i projekty powstają w wyniku pracy samoorganizujących się zespołów.
- W regularnych odstępach czasu zespół dokonuje refleksji odnośnie podnoszenia swojej efektywności, w rezultacie dostrajając i dostosowując swoje zachowania.

*http://www.agilemanifesto.org/principles.html

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



co znaczy „scrum”?

1. **scrum** (pol. tłok, ścisk) – duża grupa osób zgromadzona na małej powierzchni (**Free Dictionary**)
2. **scrum** (pol. młyn) – formacja w grze w rugby używana do wznawiania gry po przerwie lub faulu (**Wikipedia**)
3. **Scrum** – metoda zarządzania projektami zgodnie z regułami agile software development (**Wikipedia**)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

Scrum jest **narzędziem** (jednym z wielu)



- Scrum bywa zaliczany do empirycznych **metodyk zarządzania** projektami, lecz...
- Za jego twórcami (Kenem Schwaberem, Jeffem Sutherlandem, Mike-iem Beedle'm) wolę nazywać go **metodą** – zbiorem praktyk i sugestii, **strukturą** wokół której organizuje się praca zespołu
- Formalnie został spisany i zastosowany po raz pierwszy w 1993 roku, jednak swoje korzenie ma w procesach **Lean Thinking** (Lean Production/Manufacturing/Development) – lata 80 i wcześniej
- Adresowany jest przede wszystkim do zespołów pracujących nad złożonymi projektami o dużym stopniu ryzyka i niepewności
- Spełnia wymogi CMM Level 3 i ISO 9001
- Obecnie jest jednym z najpowszechniej (często łącznie z eXtreme Programming) stosowanych sposobów realizacji podejścia **Agile Software Development**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



Ujęcie historyczne

Analogia do rugby i termin „scrum” zostały zastosowane w 1986 przez Hirotaka Takeuchi i Ikujiro Nonaka w artykule „The New New Product Development Game” (Harvard Business Review) opisującym holistyczne (zintegrowane, całościowe) podejście do produkcji na podstawie badań procesów produkcyjnych w Fuji Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox i Hewlett-Packard.

Inne metody czerpiące z tego samego źródła

Dynamic System Development Method (Dane Faulkner), Adaptive Software Development (Jim Highsmith), Crystal (Alistair Cockburn), eXtreme Programming (Kent Beck, Ron Jeffries), Lean Software Development (Mary Poppendieck), Feature Driven Development (Jeff DeLuca), Behaviour Driven Developmet (Dan North)...

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

w czym może pomóc Scrum?

- poprawa produktywności* dzięki wprowadzeniu ograniczeń czasowych i wymuszeniu wartościowania prac → uporządkowanie – priorytetyzowanie i organizacja pracy
- skrócenie *time-to-market*
- zwiększenie zbieżności produktu z potrzebami (wymaganiami i oczekiwaniami) klienta → wyższa wartość wytwarzanego oprogramowania**
- poprawa jakości***
- przełamanie schematyczności myślenia → pobudzenie innowacyjności („nie da się" → „możemy ... potrzebujemy tylko ...")
- tworzenie kultury pracy zespołowej
- budowanie organizacji uczącej się (*continuous improvement*)
- zwiększanie zaangażowania pracowników → wspólny, klarowny cel; lepsza komunikacja

*nawet kilkukrotnie (źródło: „Origins of Scrum" Jeff Sutherland)
**Dr. Dobb's Journal 5/2008 Agile Adoption Survey
***tylko w połączeniu z wysoką kulturą inżynierską

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

czego nie daje?

- **nie jest magiczną różdżką,** nie leczy arogancji, ignorancji, lenistwa i wiary w cuda – powoduje natomiast, że stają się one bardzo, bardzo widoczne

- Scrum **nie jest metodyką** która mówi jak wytwarzać lepsze produkty
- Scrum **nie odpowiada** na pytanie jak należy produkować oprogramowanie wyższej jakości, lepiej, szybciej
- Scrum (jak wszystkie lekkie metody) bazuje na postulacie, że **nie istnieje meta-rozwiązanie** dla produkcji oprogramowania...

- **...możliwe jest za to zdefiniowanie ram w obrębie których stosowane procesy i narzędzia będą empirycznie doskonalone**
- **Scrum jest narzędziem, które pomaga ustalić co jest potrzebne aby produkować oprogramowanie o wyższej jakości, lepiej, szybciej**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

jak to działa?

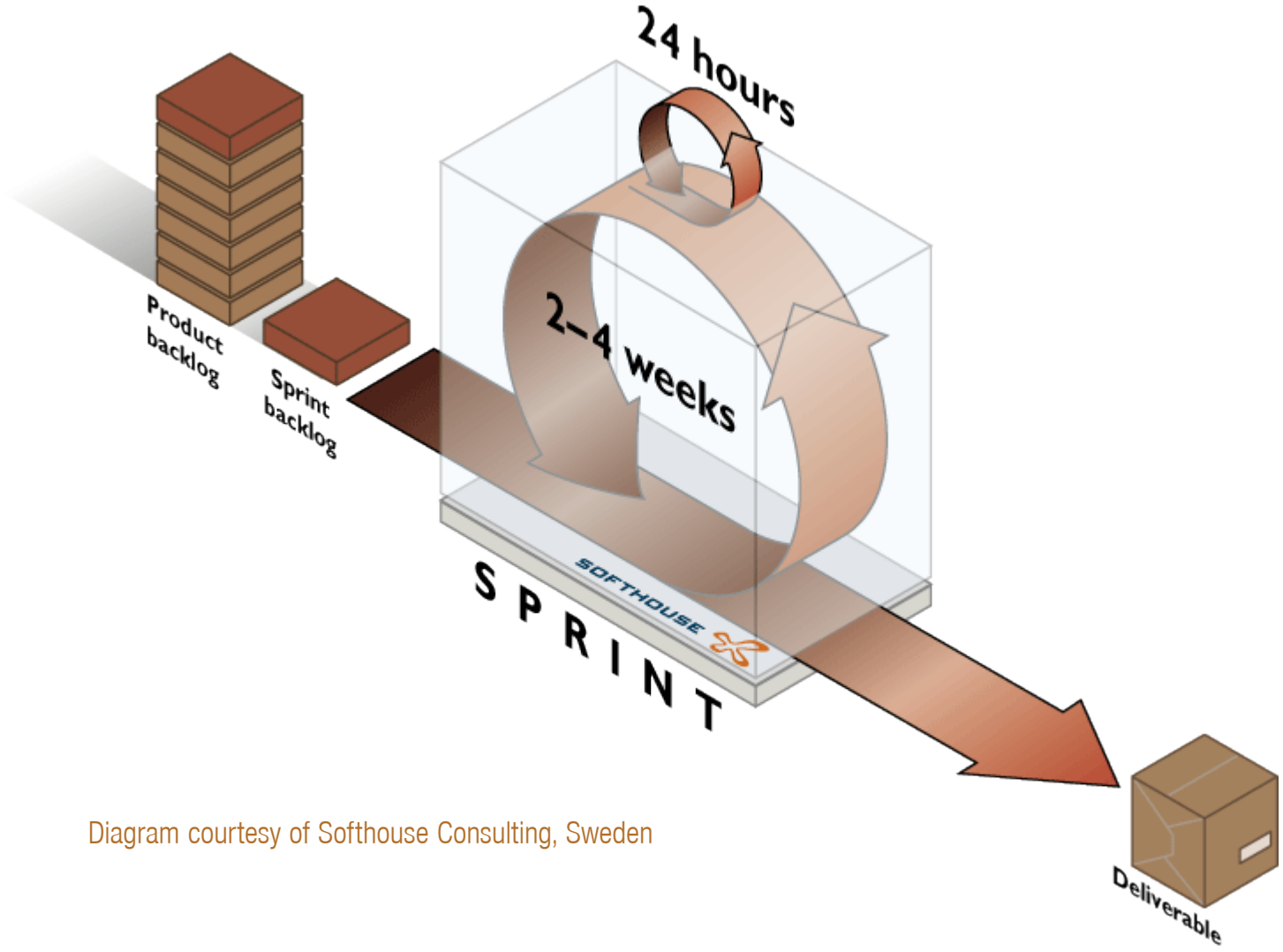


Diagram courtesy of Softhouse Consulting, Sweden

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

klarowne role i zakresy odpowiedzialności

- zespół scrumowy (**Scrum Team**)
  - **Product Owner** – (właściciel produktu) klient lub reprezentant klienta
  - **The Team** – (zespół projektowy) samoorganizująca się, interdyscyplinarna grupa specjalistów
  - **Scrum Master** – bywa nazywany liderem/przywódcą* zespołu, „młynarzem", „mistrzem", „scrum majstrem" lub „strażnikiem młyna"
- interesariusze (**Stakeholders**) – osoby spoza zespołu scrumowego zainteresowane przebiegiem prac lub od niego zależne
- **kodeks wartości:** otwartość, zainteresowanie i uwaga, poczucie odpowiedzialności, odwaga, szacunek, przejrzystość

*faktyczne przywództwo jest w Scrumie dzielone pomiędzy osobami PO a SM

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

minimalistyczne narzędzia

- rejestr (lista) prac pozostałych do wykonania (Product Backlog) – **Rejestr Produktowy** – w posiadaniu właściciela produktu, reprezentuje poziom strategiczny planowania
- rejestr (lista) zadań do wykonania w pojedynczym przebiegu (Sprint Backlog) – **Rejestr Zadaniowy** – w posiadaniu zespołu, reprezentuje poziom taktyczny planowania
- rejestr (lista) blokad (Impediment Backlog) – **Rejestr Blokad** – w posiadaniu lidera zespołu
- **definicja ukończenia** (Definition of Done, DoD)
- wykres „spalania” (Burndown Chart)

„posiadanie” nie oznacza tu wyłączności, a „jedynie” **odpowiedzialność** za utrzymanie listy, nadrzędną wartością jest **przejrzystość**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

właściciel produktu (Product Owner) definiuje wymagania i wartość produktu

- osoba odpowiedzialna za wizję produktu i efektywne przekazywanie jej zespołowi
- jest odpowiedzialna za zyskowność przedsięwzięcia
- jest odpowiedzialna za komunikację na linii biznes-development
- decyduje o kolejności implementowania funkcjonalności zgodnie z potrzebami rynku
- waliduje rezultaty pracy zespołu (proces walidacji odpowiada na pytanie czy zbudowaliśmy właściwy produkt, podczas gdy weryfikacja bada czy zbudowaliśmy ten produkt właściwie)
- jest odpowiedzialna za utrzymanie Product Backlog-u
- decyduje czy i kiedy wypuszczać produkt do klienta
- decyduje czy kontynuować projekt
- **ma ostateczny głos w kwestii priorytetów**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

My prrrecious!



(zaangażowany Product Owner)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

interesariusze projektu ustanawiają uwarunkowania i ograniczenia projektowe

- osoby, które pośrednio albo bezpośrednio, pozytywnie bądź negatywnie wpływają na projekt lub podlegają jego wpływowi...
- ...definiują uwarunkowania, ograniczenia i zależności projektowe (jawne/niejawne, świadomie/nieświadomie)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

ScrumMaster zapewnia efektywność i produktywność

- zapewnia maksymalną produktywność zespołu usuwając napotkane przez zespół przeszkody (blokady)
- zapewnia ścisłą współpracę pomiędzy reprezentantami wszystkich funkcji i ról w zespole, a także poza nim (facylitacja)
- osłania zespół przed zewnętrznymi czynnikami mogącymi zaburzyć pracę
- odpowiada za postępowanie zgodnie z praktykami Scrum
- rozważa procesy zachodzące w zespole na metapoziomie
- wspomaga proces zmian zachodzących w organizacji (każda zmiana powoduje stres ze względu na pojawiające się braki organizacji, kompetencji etc.)
- **ScrumMaster pomaga zespołowi i organizacji przezwyciężyć problemy obnażone przy pomocy Scruma**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

dobry ScrumMaster stawia wyzwania i pomaga w ich realizacji

- raczej zadaje pytania i słucha uważnie, niż udziela odpowiedzi
- wskazuje zespołowi problemy, być może sugerując sposoby i metody ich rozwiązywania (na pewno nie decyduje i nie narzuca)
- prowadzi ustawiczny *coaching* zespołu
- prowadzi spotkania i dyskusje zespołowe dążąc do konsensusu – sprawia by ludzie komunikowali się ze sobą
- dba o dobrą atmosferę w zespole
- stosuje (szczególnie we wczesnych etapach) zasadę „trust but verify”*

*ulubiona sentencja Ronalda Reagana, na podstawie przysłowia rosyjskiego

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

zespół projektowy



- niewielka, **samoorganizująca się**, **interdyscyplinarna**, a przez to w dużej mierze **samowystarczalna** grupa (orientacyjnie 7±2 osób)
- ma prawo (i obowiązek) **samodzielnie podejmować decyzje** prowadzące do spełnienia celu sprintu
- grupa powinna być **jednorodna** pod względem hierarchii służbowej
- grupa powinna być **jednolita** pod względem składu na cały czas trwania projektu (ewentualne zmiany tylko pomiędzy przebiegami)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

rejestr produktowy (Product Backlog)

- **jawna** (dostępna na wszystkich), **spriorytetyzowana** lista prac pozostałych do wykonania przez zespół w ramach produktu, projektu czy release'u
- reprezentuje **potrzeby** (wymagania i oczekiwania) klienta (*requirements* vs *desirements*)
- zbiera wymagania funkcjonalne, niefunkcjonalne, defekty, pomysły na przyszłość etc.
- elementy listy (*backlog items*) mogą być uzupełniane przez dodatkową dokumentację (reguły biznesowe, przykłady formularzy, akty prawne, przypadki użycia, scenariusze i przykłady wejść i wyjść z systemu etc.)
- każdy element listy **musi być oszacowany**
- **wszyscy mogą dodawać elementy, ale tylko Product Owner ma prawo nadawać im priorytety (wartość)**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

rejestr produktowy (Product Backlog)

- elementy wędrując w górę listy nabierają szczegółowości, rozpadają się na drobniejsze wpisy o rozmiarze pozwalającym zespołowi na zrealizowanie kilku-kilkunastu w jednym sprincie (2-3 dni dla 2-3 osób na jeden element)
- do rozpoczęcia pracy niezbędne jest przygotowanie Rejestru Produktowego w **rozsądnym** zakresie → planowanie *just-in-time* (zwykle 2-3 sprinty w przód, ale nierzadko kilkanaście sprintów)
- często jawnie zaznaczana jest granica wyznaczona przez daty wydań (*release*) lub datę końcową projektu (*waterline*) → **prędkość zespołu**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

cykl produkcyjny (Sprint)

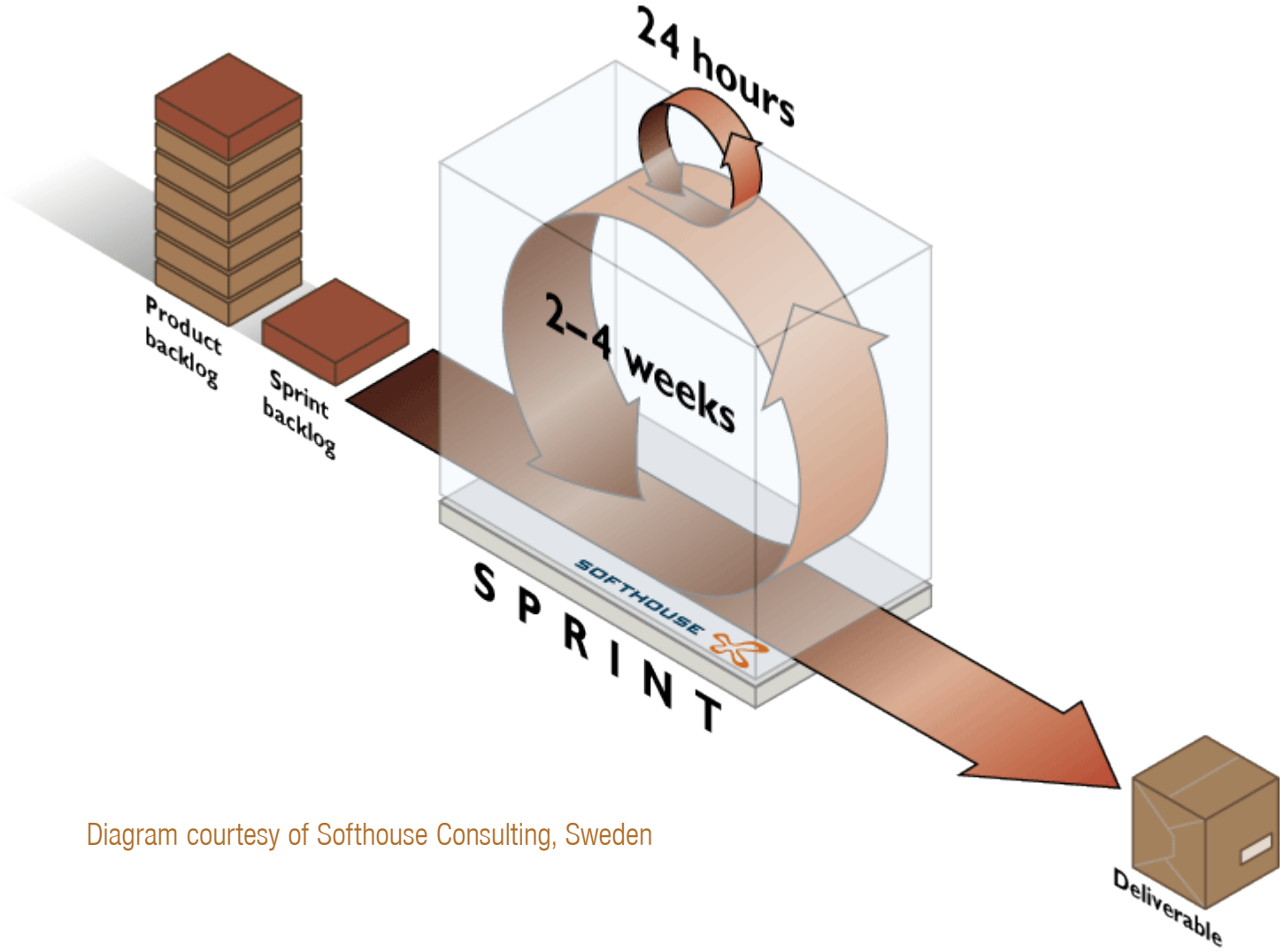


Diagram courtesy of Softhouse Consulting, Sweden

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek, Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

planowanie sprintu (Sprint Planning)



Cel sesji planistycznych:

- przekazanie wizji i priorytetów
- konfrontacja wizji i wykonalności
- zdefiniowanie celu sprintu odzwierciedlającego zobowiązanie przyjęte przez zespół
- zdefiniowanie zadań i szacowanie (konstruowany jest rejestr zadaniowy – Sprint Backlog)

Bierzemy pod uwagę:

- możliwości zespołu
- aktualny stan produktu
- funkcjonalność pozostająca w rejestrze produktowym
- warunki biznesowe i priorytety
- obecnie wykorzystywane a dostępne technologie

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

kawałek po kawałku...

mały, ale kompletny wycinek



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

koncepcja kawałka tortu lub „kroczącego szkieletu”

- za każdym razem wytwarzany jest pionowy wycinek systemu
- obsługujący małą część wymagań, ale w sposób kompletny („end-to-end”)
- w większości przypadków architektura systemu może ewoluować wraz z funkcjonalnością (refaktoryzacja, ciągła integracja i testowanie)
- nie należy tracić czasu na zbędny perfekcjonizm (*gold-plating*) i popadać w paraliż analityczny (*analysis-paralysis*) (pracujemy z nieprecyzyjnymi wymaganiami, co sprint weryfikując z klientem kierunek rozwoju produktu)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



rejestr zadaniowy

- zespół samodzielnie decyduje jakie zadania należy wykonać aby osiągnąć **cel sprintu**
- zespół samodzielnie organizuje swoją pracę tak, aby wypełnić powzięte **zobowiązanie**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

rejestr zadaniowy (Sprint Backlog)

- lista zadań, które należy wykonać aby przetworzyć wybrane elementy rejestru produktowego w potencjalnie zbywalny produkt
- zadania zwykle szacowane są w idealnych godzinach (<16h)
- większe zadania (niewiadome podczas planowania) powinny być dekomponowane na mniejsze w trakcie trwania sprintu – **zanim przystąpi się do pracy!**
- zadania nie są przypisywane do członków zespołu podczas sesji planowania, zespół samodzielnie decyduje o tym kto, co i kiedy wykona w trakcie sprintu
- często rejestr zadaniowy odzwierciedla schemat przepływu pracy przez zespół/stan realizacji

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

przebieg sprintu

2-4 tygodni

- zakres prac jest zamrażany
- zespół samodzielnie wypracowuje najlepszą drogę do wypełnienia celu iteracji
- zespół jest izolowany od czynników zewnętrznych mogących zakłócić pracę
- właściciel produktu jest dostępny dla zespołu aby odpowiadać na pytania i wątpliwości
- Scrum Master czuwa nad zapewnieniem maksymalnej produktywności zespołu

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



wyzwanie stojące przed zespołem –

dostarczenie kompletnego, w pełni działającego, **potencjalnie zbywalnego przyrostu produktu** zgodnie z podjętymi zobowiązaniami – celem sprintu i kryterium ukończenia (DoD)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



samoorganizacja

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

samoorganizacja w Scrumie

- członkowie zespołu sami decydują o tym jakie zadania należy wykonać aby wypełnić cel sprintu i dopełnić definicji gotowości produkcyjnej (**Definition of Done**)
- członkowie zespołu przypisują sobie zadania samodzielnie – kolejno (sekwencyjnie), w wybranym przez siebie porządku
- każdy członek zespołu może dodawać, usuwać i modyfikować elementy rejestru zadaniowego – tak długo, jak pozostaje to w zgodzie z **celem sprintu** – informując o tym fakcie pozostałych
- codziennie szacowana jest liczba godzin pozostałych do zakończenia rozpoczętych zadań – **przejrzystość**
- analiza stanu prac i podejmowanie decyzji co do kolejnego ruchu dokonywane jest w oparciu o wykres spalania (**Sprint Burndown Chart**)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



15 minut **codziennie** - codzienny scrum (stand-up meeting)

puls zespołu

> U podstaw wszystkich problemów z projektami leży fakt, że ktoś komuś o czymś ważnym nie powiedział.
>
> **–Kent Beck**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

puls zespołu – 3-6 pytań skierowanych do każdego członka zespołu

- Jakie zadania **zakończyłeś** od ostatniego spotkania?
- Nad jakimi zadaniami **będziesz pracować** do następnego spotkania?
- Czy jest coś co cię **blokuje** i/lub może **zagrażać osiągnięciu celu** sprintu?

- Jak oceniasz (w skali 1-10) swoją pewność, że dotrzymamy celu sprintu?
- Czy jest coś co powinniśmy dodać do listy zadań?
- Czy dowiedziałeś się/nauczyłeś się czegoś o czym powinni wiedzieć inni członkowie zespołu?

Larman, Craig. Agile and Iterative Development: A Manager's Guide. Addison-Wesley (2004)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

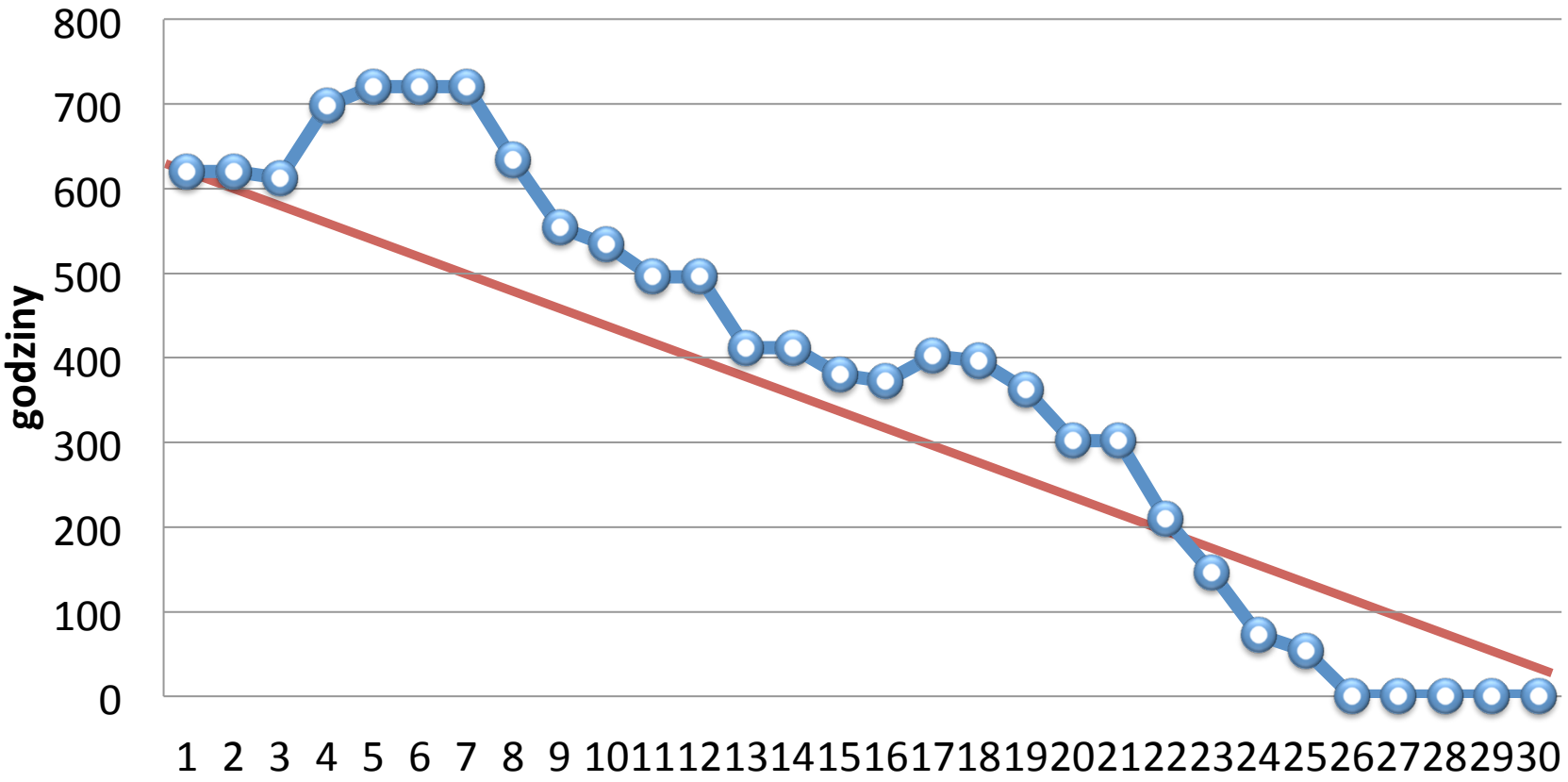


- narzędzie monitorowania ilości pracy pozostałej do wykonania
- wspiera dążenie do celu – pomaga utrzymać odpowiedni kierunek i tempo prac
- wpływa na morale zespołu
- analiza trendów → przeszkody, nieplanowane zadania, ryzyka i zależności projektowe

wykres spalania

© 2006–2011 Tomasz Włodarek, Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

wykres typu „Sprint Burndown”



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

rejestr blokad (Impediments Backlog)

- każda sytuacja wynikła w trakcie prac, która opóźnia lub całkowicie wstrzymuje pracę, a z którą zespół nie jest w stanie sobie poradzić samodzielnie jest **blokadą**
- każdy członek zespołu ma **obowiązek** komunikować tego typu sytuacje (na codziennych scrumach lub częściej)
- ScrumMaster zarządza rejestrem blokad (ustala z zespołem priorytety, aktualizuje status)
- używając dostępnych mu mechanizmów organizacyjnych (w tym eskalacji) usuwa blokady oraz...
- ...raportuje stan do zespołu (również na codziennych scrumach, odpowiadając na te same trzy pytania)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

zamknięcie sprintu (Sprint Review, Sprint Retrospective)

4 + 3 godziny

- **przegląd sprintu** – prezentacja wyników pracy zespołu
- dyskutujemy spostrzeżenia co do wykonanej pracy, wyciągamy wnioski na kolejny sprint
- nie zastępuje bram jakości
- wszyscy są zaproszeni, także osoby spoza zespołu

- **retrospektywa** – refleksja zespołu:
  - co poszło dobrze?
  - co poszło źle?
  - co chcemy zacząć robić?
  - co robimy niepotrzebnie?
  - jak zmienić to co nas boli?

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

przegląd sprintu (Sprint Review)

- zespół prezentuje ukończoną (spełniającą kryteria → **Definition of Done**), działającą, **potencjalnie zbywalną** funkcjonalność
- niedopuszczalna jest „prawie" skończona praca (dokumentacja bez kodu, niedokończony kod, nieudokumentowany kod, nieskompilowany kod, nieuruchomiony kod, nieprzetestowany kod, kod zawierający defekty odłożone na „później", to-ciekawe-bo-u-mnie-działa itd.)
- prezentacja odbywa się dla i z perspektywy klienta, a nie programisty – odnosimy się do celu sprintu, osiągniętej wartości biznesowej
- pozostajemy otwarci na pytania i krytykę, dyskutujemy z właścicielem produktu spostrzeżenia → przegląd sprintu to **sesja robocza** przygotowująca grunt pod kolejne planowanie (uwaga na znaczenie słowa „demo")
- wyciągamy wnioski na kolejny sprint (→ zmiany w Product Backlogu)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

jak wykorzystujecie Scruma do znajdowania i rozwiązywania problemów?

analiza i ...

> Niezależnie co odkryjemy, musimy żywić przekonanie, że wszyscy wykonali swoją pracę tak dobrze jak to tylko możliwe, biorąc pod uwagę ich umiejętności, możliwości, dostępne zasoby i aktualną sytuację.
>
> **–Norman Kerth**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

jak wykorzystujecie Scruma do znajdowania i rozwiązywania problemów?

...adaptacja

> W podejściu holistycznym (całościowym), w odróżnieniu od postępowania zgodnie ze zdefiniowanymi etapami (podejście liniowe), proces produkcji wyłania się z nieustannej interakcji członków interdyscyplinarnego zespołu.
>
> **–Hirotaka Takeuchi, Ikujiro Nonaka**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

retrospektywa (Sprint Retrospective)

- spotkanie mające na celu ustalenie co poszło dobrze, a co nie, przedyskutowanie czego się dowiedzieliśmy oraz wyciągnięcie wniosków i przygotowanie akcji naprawczych
- dyskutowane są procesy, procedury, narzędzia, współpraca w grupie, środowisko pracy
- retrospektywy odbywają się regularnie w trakcie trwania projektu, po każdym sprincie
- bierze udział cały zespół – reprezentowane są wszystkie perspektywy (deweloperzy, QA, marketing, customer support, etc.)
- mówimy za siebie i ze swojej perspektywy, nie przerywamy innym
- jeśli trzeba wprowadzamy „safety check” – pytanie o komfort i swobodę wypowiedzi

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

monitorowanie stanu projektu



gdzie jesteśmy...?

- **Working Software** – podstawowa miara stanu zaawansowania projektu → **Running Tested Features** (uruchomiona, przetestowana i... odebrana? funkcjonalność – Ron Jeffries)
- **Sprint Burndown Chart** – narzędzie oceny stanu prac w iteracji
- **Velocity** – podstawowa charakterystyka zespołu
- **Release Burndown Chart** – narzędzie oceny stanu wykonalności projektu

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

prędkość zespołu (Team Velocity)

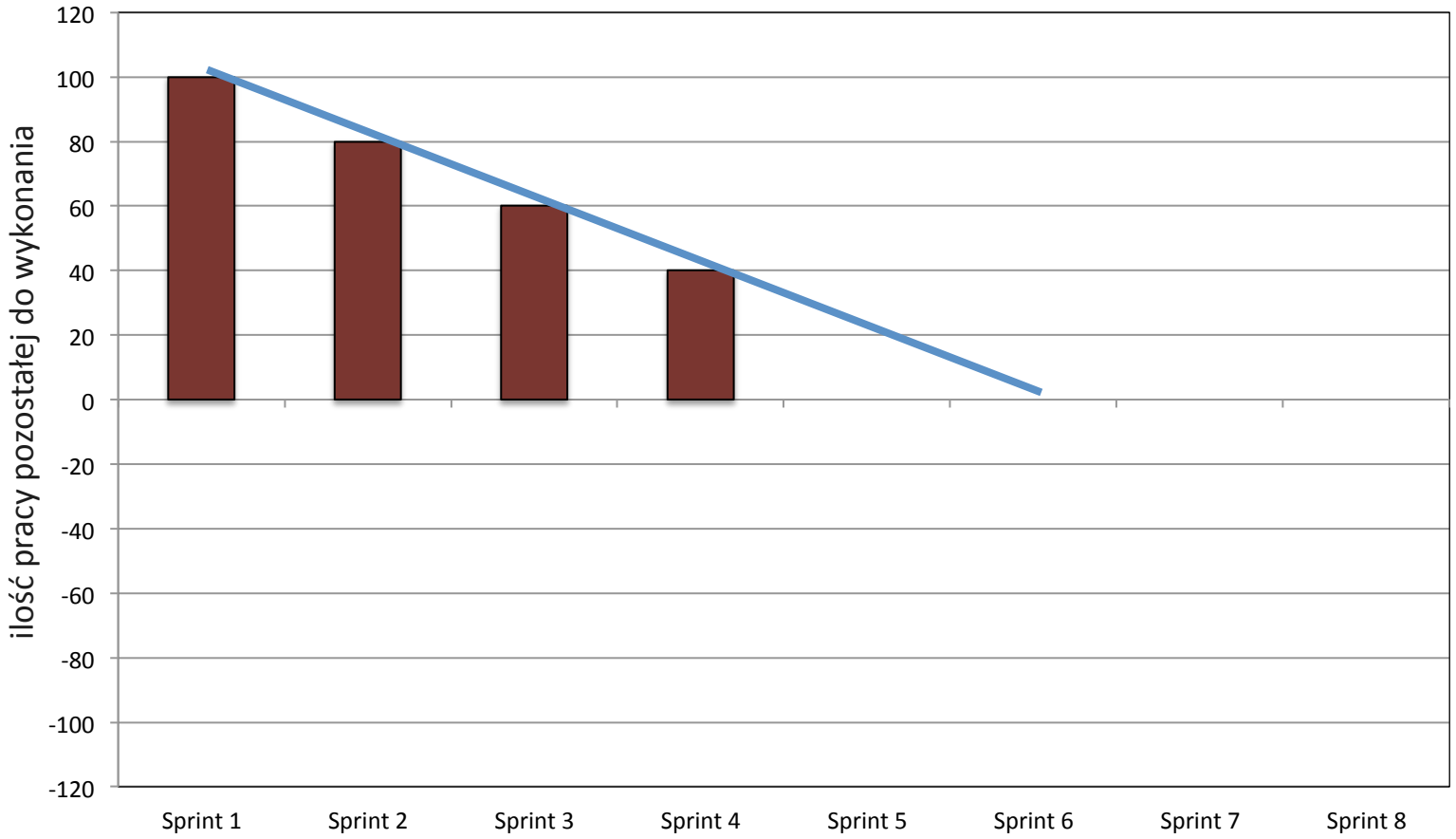
- zdolność produkcyjna zespołu
- liczba punktów (jednostki wg której szacowane są elementy rejestru produktowego), które zespół przetwarza w potencjalnie zbywalny produkt w trakcie trwania jednego sprintu
- prędkość będzie zmieniać się ze sprintu na sprint, przy stałym zespole powinna ustabilizować się po kilku (3-6) sprintach
- na potrzeby planowania z większym wyprzedzeniem niż jeden sprint można użyć ostatniej, średniej, minimalnej lub maksymalnej wartości

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

wykres typu „Release Burndown”



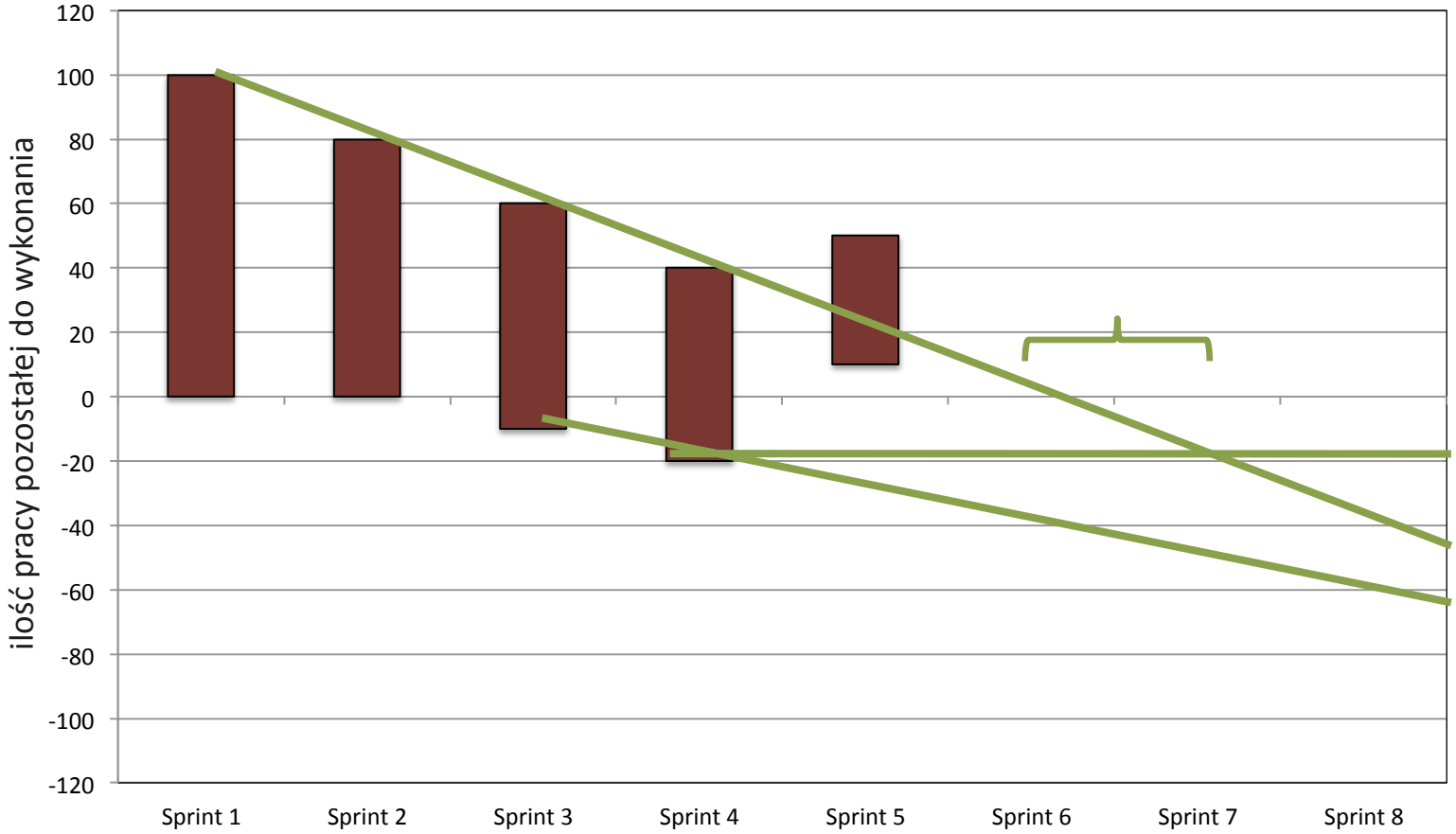
© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

wykres typu „Release Burndown”

- za każdym razem gdy etap prac (historie użytkownika, cechy produktu) jest kończony, góra słupka jest obniżana
- za każdym razem gdy praca jest powtórnie szacowana (re-estymowana) góra słupka przesuwana jest w górę (zwiększenie szacunków) lub w dół (obniżenie)
- za każdym razem gdy zakres prac jest zwiększany, dół słupka jest obniżany
- za każdym razem gdy zakres prac jest zmniejszany, dół słupka jest podnoszony
- **zawsze gdy dół słupka znalazł się poniżej linii 0 początkowy zakres prac został zwiększony**

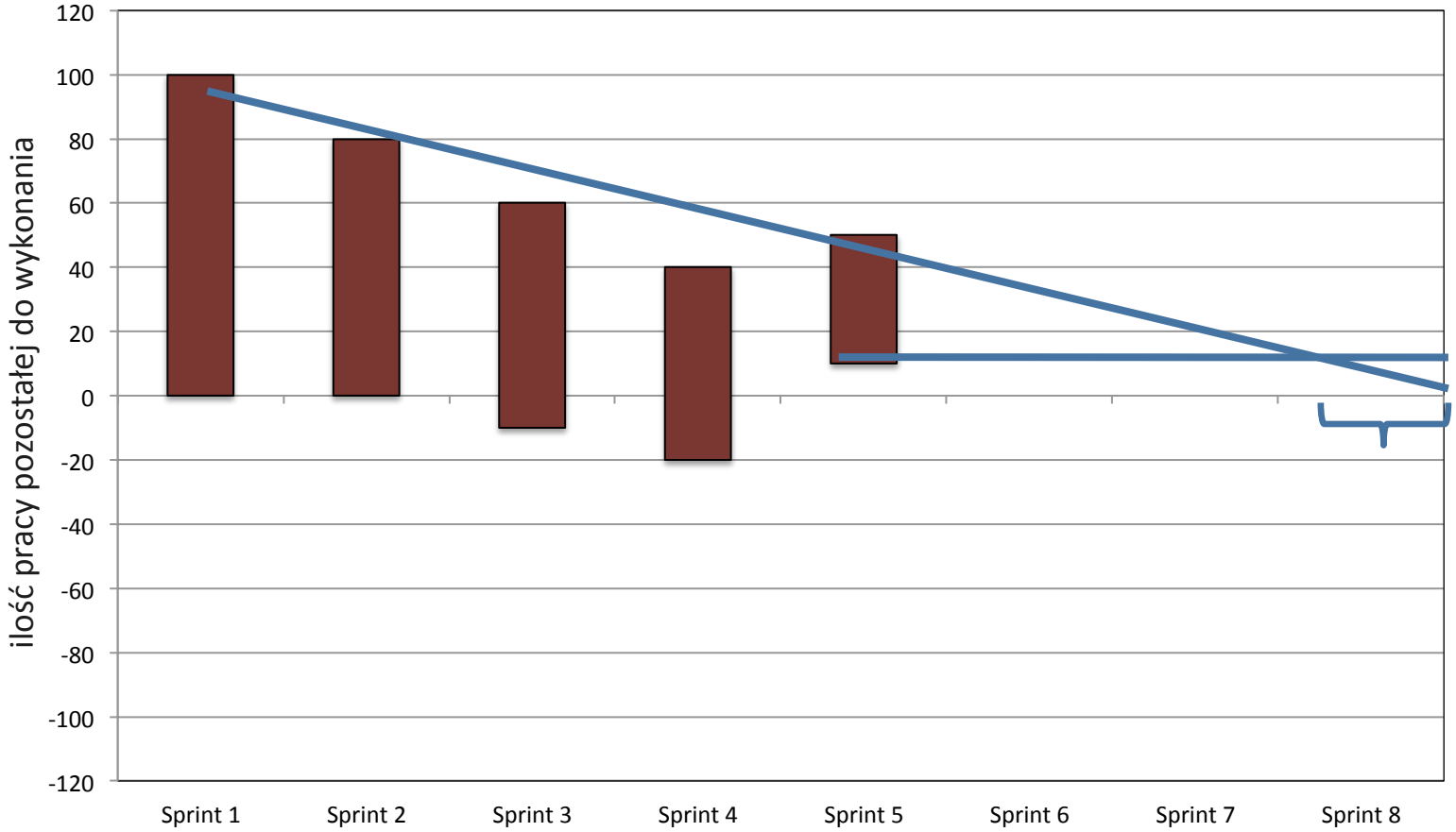
© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

wykres typu „Release Burndown”



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

wykres typu „Release Burndown”



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

scrum jest trudny

> Młyn (scrum) to najbardziej niebezpieczna faza gry w rugby. Zapadnięcie się młyna lub nieuważne postępowanie może prowadzić do kontuzji kręgosłupa lub nawet śmierci (przez skręcenie karku) zawodnika w pierwszej linii młyna.
>
> **–Wikipedia**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

proste lecz...

rygorystyczne reguły

> Zawsze gdy ktoś chce usunąć jeden z podstawowych elementów Scrum'a, robi tak ponieważ element ten obnaża aspekty rzeczywistości których nikt nie chce zauważać.
>
> **–Ken Schwaber**

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

Scrum czy Scrum, but?



© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

konieczna zmiana sposobu myślenia i...

odpowiedzialność zespołu

nie antycypujemy kształtu produktu podczas długotrwałego, wstępnego planowania, pozwalamy mu ewoluować obniżając koszty i skracając czas wytworzenia dopełniających się (lub alternatywnych) rozwiązań



© 2006–2011 Tomasz Włodarek, Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

zagadnienie zaawansowane



- planowanie produkcji
- ustalanie czasu trwania sprintu
- praca z wymaganiami/historiami użytkownika (definiowanie, łączenie w tematy i opowieści, ustalanie kolejności)
- szacowanie, techniki szacowania
- łączenie ról
- przypisania cząstkowe
- przedterminowe zakończenie iteracji
- sprint zerowy
- sprint stabilizacyjny

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.

zagadnienia złożone



- rodzaje kontraktów
- projekty dużej skali, skalowalność Scruma
- synchronizacja zespołów projektowych
- łączenie Scruma z procesami kaskadowymi
- outsourcing i off-shoring
- środowisko pracy, aranżacja przestrzeni biurowej

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.


dziękuję!



tomek@poddrzewem.pl
http://www.linkedin.com/in/wlodarek

http://www.poddrzewem.pl
http://www.scrum.org
http://www.scrum.org.pl
http://www.scrumalliance.com
http://www.agilealliance.com
http://www.controlchaos.com
http://www.mountaingoatsoftware.com/
http://groups.yahoo.com/group/scrumdevelopment/

Przewodnik po metodzie Scrum, Ken Schaber, Jeff Sutherland, 2010
Death March, Edward Yourdon, Prentice Hall, 2004
The New New Product Development Game, Hirotaka Takeuchi, Ikujiro Nonaka, Harvard Business Review, Jan-Feb 1986
Toyota Production System: Beyond Large-Scale Production, Taiichi Ohno
Lean Software Development: An Agile Toolkit, Mary Poppendieck, Tom Poppendieck
Agile and Iterative Development: A Manager's Guide, Craig Larman
Scaling Lean & Agile Development, Craig Larman, Bas Vodde
Agile Project Management: Creating Innovative Products, Jim Highsmith
Agile Project Management with Scrum, Ken Schwaber
The Enterprise and Scrum, Ken Schwaber
Agile Estimating and Planning, Mike Cohn, Prentice Hall, 2006
User Stories Applied, Mike Cohn, Addison Wesley, 2004
Agile Retrospectives: Making Good Teams Great, Esther Derby, Diana Larsen, Pragmatic Bookshelf, 2006
Fearless Change: Patterns for Introducing New Ideas, Manns/Rising, Addison-Wesley, 2004
Software by Numbers: Low-Risk, High-Return Development, Mark Denne, Jane Cleland-Huang, Prentice Hall, 2004
Succeeding with Agile, Mike Cohn, Addison-Wesley, 2010


klinika
lekkich
przypadków
http://www.poddrzewem.pl/klinika



1. Jak liczny jest zespół
2. Jaka jest struktura organizacyjna zespołu (kompetencje, stanowiska służbowe, role)
3. Czy zespół jest/może być samowystarczalny, czy też występuje konieczność synchronizowania prac z innymi zespołami/specjalistami/podwykonawcami
4. Czy członkowie zespołu mają doświadczenie we wspólnej pracy
5. Jakie doświadczenie ma zespół w domenie w której realizowane będą projekty
6. Jakie doświadczenie ma zespół w technologii w której realizowane będą projekty
7. Jakie doświadczenie ma zespół w szeroko rozumianej inżynierii oprogramowania
8. Kto jest odbiorcą oprogramowania, a kto jego użytkownikiem
9. Kto jest w stanie ich (odbiorców/użytkowników) reprezentować i podejmować wiążące w tym zakresie decyzje, a kto jedynie dostarcza informacji
10. Jakiego rodzaju projekty będą przedmiotem prac (nowy produkt, utrzymanie istniejącego kodu, rozszerzanie istniejącego produktu)
11. Jaka jest natura realizowanego przedsięwzięcia (oprogramowanie na zamówienie, na półkę, dostępne w sieci, zamknięte systemy, ...)
12. Jakie są ograniczenia i uwarunkowania projektowe (czas, budżet), jakie są ryzyka projektowe
13. Czy do tej pory stosowany był zdefiniowany proces produkcji (jaki? z jakimi praktykami/procedurami zespół jest zaznajomiony?)
14. Czy były stosowane narzędzia (software) do zarządzania realizowanymi projektami (jakie?)

© 2006–2011 Tomasz Włodarek. Pragmatyczne metody wytwarzania oprogramowania. Wszystkie prawa zastrzeżone.